Lab 5

(i)169

class Solution {

    public int majorityElement(int[] nums) {

     int n = nums.length;

        Arrays.sort(nums);

        return nums[n/2];

    }

(ii)2706

class Solution {

    public int buyChoco(int[] prices, int money) {

        Arrays.sort(prices);

      return prices[0] + prices[1] > money ? money : money - (prices[0] + prices[1]);

    }

}

(iii)2733

class Solution {

    public int findNonMinOrMax(int[] nums) {

        if(nums.length<=2)

            return -1;

        else

           return Math.max(Math.min(Math.max(nums[0], nums[1]), nums[2]), Math.min(nums[0], nums[1]));

    }

}

(iv)268

class Solution {

    public int missingNumber(int[] nums) {

        Arrays.sort(nums);

         for(int i=0;i<nums.length;i++)

         {

             if(nums[i]!=i)

                return i;

         }

         return nums.length;

    }

}

(v)2160

class Solution {

    public int minimumSum(int num) {

     int[] nums = new int[4];

        int i=0;

        while(num>0){

            nums[i]=num%10;

            num=num/10;

            i++;

        }

        Arrays.sort(nums);

        return nums[0]\*10+nums[3]+nums[1]\*10+nums[2];

    }

}

(vi)1200

class Solution {

    public List<List<Integer>> minimumAbsDifference(int[] arr) {

     Arrays.sort(arr);

        int diff = Integer.MAX\_VALUE;

        for(int i=1;i<arr.length;i++){

            if(arr[i]-arr[i-1]<diff)

                diff=arr[i]-arr[i-1];

        }

        List<List<Integer>> list = new ArrayList();

        for(int j=1;j<arr.length;j++){

            if(arr[j]-arr[j-1]==diff){

                ArrayList<Integer> l = new ArrayList<>();

                l.add(arr[j-1]);

                l.add(arr[j]);

                list.add(l);

            }

        }

        return list;

    }

}

(vii)561

class Solution {

    public int arrayPairSum(int[] nums) {

       Arrays.sort(nums);

        int sum=0;

        for(int i=0;i<nums.length;i+=2)

        {

          sum+=nums[i];

        }

        return sum;

    }

}

(viii)1122

class Solution {

    public int[] relativeSortArray(int[] arr1, int[] arr2) {

        int arr[] = new int[1100];

        int k=0;

        int a1[] = new int[arr1.length];

        int l=0;

        for (int i=0; i<arr.length; i++){

            if (k< arr1.length){

                int ascii = (int)arr1[k];

                arr[ascii]+=1;

                k++;

            }

        }

        for (int i=0; i<arr2.length; i++){

            int a=arr[arr2[i]];

            for (int j=0; j<a; j++){

              a1[l]=arr2[i];

              l++;

              arr[arr2[i]]=0;

            }

        }

        for (int i=0; i< arr.length; i++){

           if (arr[i]>0){

              int a = arr[i];

              for (int w=0; w<a; w++){

                  a1[l]=i;

                  l++;

              }

           }

       }

       return a1;

    }

}

(ix)147

class Solution {

public ListNode insertionSortList(ListNode head) {

if(head == null && head.next == null) return head;

ListNode dummy = new ListNode(-1, head);

ListNode curr = head;

ListNode numInsert, prev;

while(curr != null && curr.next != null) {

if(curr.val <= curr.next.val) {

curr = curr.next;

} else {

numInsert = curr.next; // numInsert -> the smaller value which needs to be its correct position

prev = dummy; // prev -> this indicate the position where numInsert will be inserted

while(prev.next.val < numInsert.val) {

prev = prev.next;

}

curr.next = numInsert.next;

numInsert.next = prev.next;

prev.next = numInsert;

}

}

return dummy.next;

}

}

(x)912

class Solution:

    def sortArray(self, nums: List[int]) -> List[int]:

        nums.sort()

        return nums

(xi)148

class Solution {

public ListNode sortList(ListNode head) {

int count = 0;

ListNode temp = head;

while(temp!=null){

count++;

temp = temp.next;

}

int[] arr = new int[count];

temp = head;

count = 0;

while(temp!=null){

arr[count++] = temp.val;

temp = temp.next;

}

Arrays.sort(arr);

temp = head;

count = 0;

while(temp!=null){

temp.val = arr[count++];

temp = temp.next;

}

return head;

}

}

(xii)75

class Solution {

  public void sortColors(int[] nums) {

    int l = 0;

    int r = nums.length - 1;

    for (int i = 0; i <= r;)

      if (nums[i] == 0)

        swap(nums, i++, l++);

      else if (nums[i] == 1)

        ++i;

      else

        swap(nums, i, r--);

  }

  private void swap(int[] nums, int i, int j) {

    final int temp = nums[i];

    nums[i] = nums[j];

    nums[j] = temp;

  }

}

(xiii)451

class Solution:

    def frequencySort(self, s: str) -> str:

        m=Counter(s)

        s1=dict(sorted(m.items(),key=lambda x:x[1],reverse=True))

        string=""

        for i,j in s1.items():

            string+=i\*j

        return string

(ixv)791

class Solution:

    def customSortString(self, order: str, s: str) -> str:

        indexMap = {char: index for index, char in enumerate(order)}

        return ''.join( sorted( s, key=lambda char: indexMap.get(char, len(s)) ) )

(xv)347

class Solution {

    public int[] topKFrequent(int[] nums, int k) {

     Map<Integer, Integer> map = new HashMap<>();

        for (int i : nums)

            map.merge(i, 1, Integer::sum);//For Getting Frequency

        List<Integer> list = new ArrayList<>(map.keySet());

        list.sort((a, b) -> map.get(b) - map.get(a)); //Sort by Frequency in descending order

        int res[] = new int[k];

        for (int i = 0; i < k; ++i)

            res[i] = list.get(i);

        return res;

    }

}

(xvi)1338

class Solution {

    public int minSetSize(int[] arr) {

        Map<Integer, Integer> map = new HashMap<>();

    ArrayList<Integer>[] list = new ArrayList[arr.length + 1];

    for (int num : arr) {

        map.put(num, map.getOrDefault(num, 0) + 1);

    }

    for (int num : map.keySet()) {

        int count = map.get(num);

        if (list[count] == null) {

            list[count] = new ArrayList<Integer>();

        }

        list[count].add(num);

    }

    int steps = 0, res = 0;

    for (int i = arr.length; i > 0; i--) {

        List<Integer> cur = list[i];

        if (cur == null || cur.size() == 0) continue;

        for (int num : cur) {

            steps += i;

            res++;

            if (steps >= arr.length / 2)

                return res;

        }

    }

    return arr.length;

}

}